Keycloak and G8: sign-in at the gateway, authorization on the Route

A Route that only serves signed-in users of a Keycloak realm. A browser is sent to Keycloak to sign in, an API client sends a bearer token, and the backend receives the user's identity in a header. Keycloak's own authorization then decides which paths a user may call. Every call, allowed or refused, is a request record, and the change that put the Plugins on the Route is in the audit log with its secrets masked. Taken from the step-by-step article at g8.dev/blog/keycloak-integration.

What you need

A Keycloak realm the G8 engine can reach
The guide uses the realm corp with a confidential client payments-gateway (standard flow, service account and Authorization on) and two users, one with the realm role finance.

A Route with a backend
payments-api serving /payments/* from the Upstream payments-backend; the backend prints the identity it finds in the X-Userinfo header.

An Environment with Traffic Explorer on
So every call becomes a request record, and a role that may manage Routes.

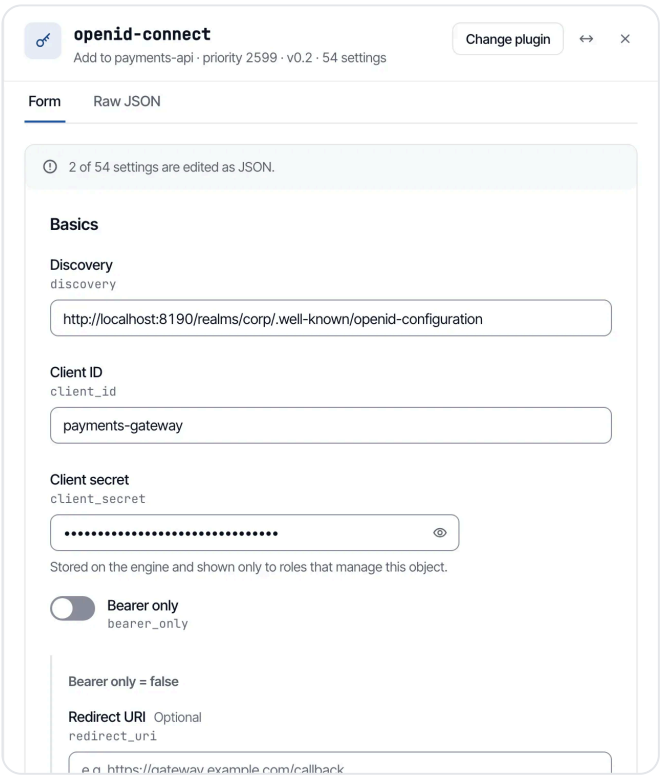
Part one: OpenID Connect on the Route

Add the Plugin
Route › Plugins › Add plugin › openid-connect. Under Basics: the realm's discovery document, the client ID and the client secret (stored on the engine, shown masked), the scope openid profile email.

Two settings outside Basics
Session › Secret signs the browser's cookie; the form refuses to save without it while Bearer only is off. Behavior › Use jwks must be on, or a bearer token in the Authorization header is ignored and the API client is sent to the sign-in page.

A browser signs in
The engine finds no session, answers 302 to Keycloak, exchanges the code for tokens on the way back and forwards the request. The backend reads the claims from X-Userinfo; it never sees a password or a token.

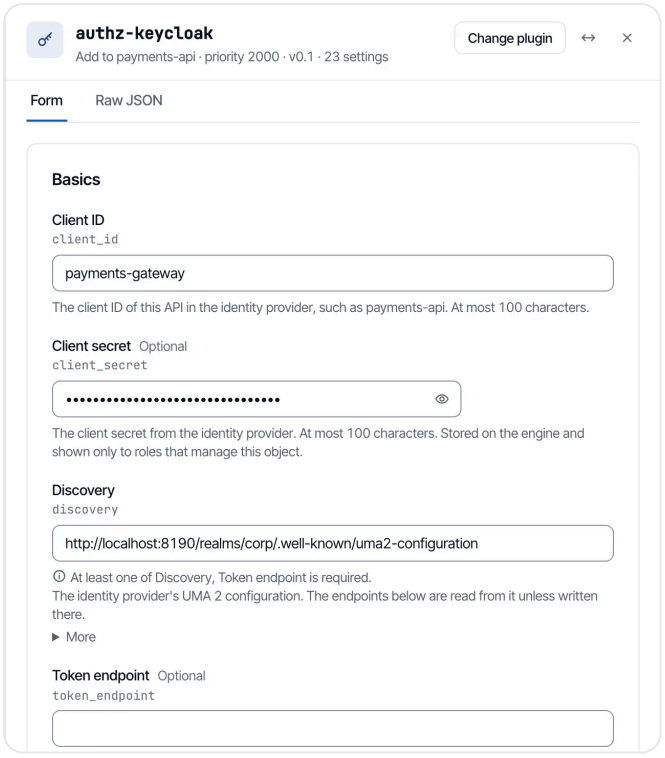
An API client sends a token
A bearer token from Keycloak's token endpoint is verified against the realm's keys; a token the engine cannot verify is refused with 401 at the gateway, before the backend is called.



The openid-connect form on the Route: the discovery URL of the realm, the client ID, the client secret already masked while you type it.

Part two: Keycloak authorization on the same Route

openid-connect answers who the caller is. Which paths that caller may use is Keycloak's decision, and the Route asks it with a second Plugin.



The authz-keycloak form: the same client, the realm's UMA 2 discovery document instead of the OpenID one.

The resources in Keycloak

In the client's Authorization tab, the resource refunds carries /payments/refunds/* and a permission that needs the role finance; the resource orders carries /payments/orders/* and a permission any signed-in user passes.

The second Plugin

authz-keycloak with the same client ID and secret, the realm's UMA 2 discovery document, and Behavior › Resolve the resources from the path switched on; the Permissions list stays empty because the path decides.

When it runs

After openid-connect, so it always sees a verified token. It asks Keycloak for a decision on the path and keeps the answer in its cache.

The last row below

Under the default Policy enforcement mode ENFORCING, a path no resource covers is refused for everyone. Register every path the Route serves, or put the paths that need no authorization on a Route of their own.

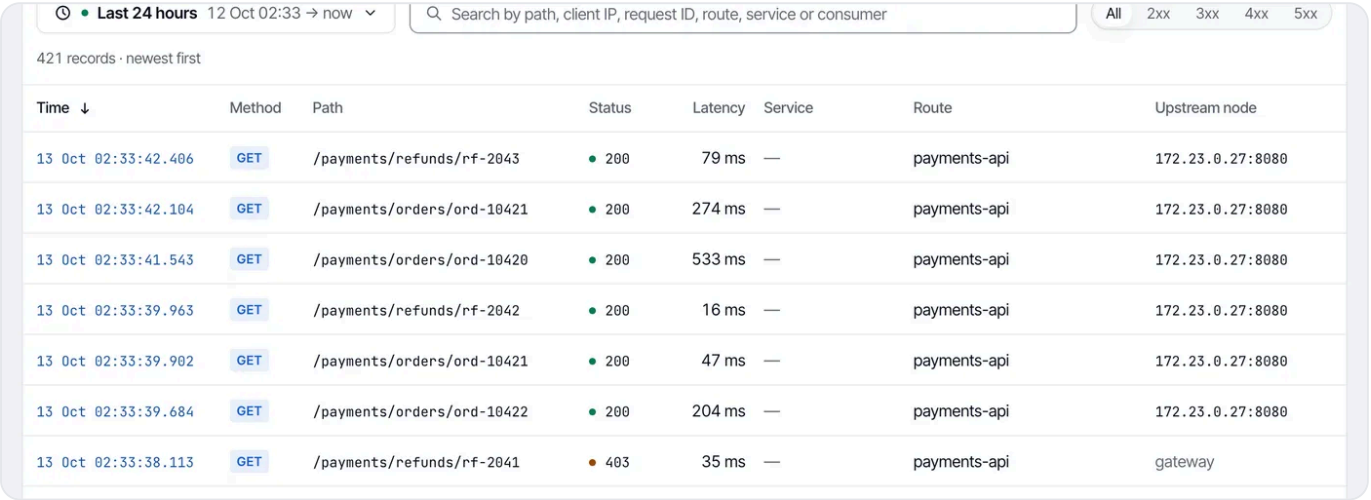
USER	REQUEST	ANSWER
jane.miller (finance)	GET /payments/refunds/rf-2041	200
jane.miller (finance)	GET /payments/orders/ord-10421	200
sam.ortiz	GET /payments/refunds/rf-2041	403 access_denied
sam.ortiz	GET /payments/orders/ord-10421	200
jane.miller (finance)	GET /payments/me	403: under the default ENFORCING mode a path no resource covers is refused for everyone

What the gateway records

Traffic Explorer
The 200s sit beside the 401 and the 403s. The Upstream node column says who answered: the backend's node for a 200, gateway for a refusal the engine made itself.

Masked headers
The authorization header is masked before the record is stored. openid-connect adds x-access-token and x-userinfo with the user's claims; add both under Traffic Explorer › Settings › Masking, and every record from then on masks them.

The audit log
Adding the Plugin was a change to the Route: who, when, the configuration before and after, with client_secret and session.secret masked on both sides.



Request records of the Route: the refusals never reached the backend; the Upstream node column reads gateway for them.

Side by side with other identity methods

A Route can admit more than one kind of caller: multi-auth takes a list of authentication Plugins and tries them in order until one admits the request; key-auth, basic-auth, jwt-auth and hmac-auth combine this way. openid-connect keeps its own session and stays a Plugin of its own. When the Route works, copy it to the production Environment; G8 shows what the copy changes before anything is written, and the copy is an audit record.

NEXT STEP

Bring your Keycloak realm and one API; we put both Plugins on a Route in the demo. Request a demo at g8.dev/request-a-demo or write to hello@g8.dev. Read next: G8 API Gateway Capabilities Reference (2 pages), at g8.dev/downloads.

g8.dev/request-a-demo · hello@g8.dev